UNITED STATES PATENT APPLICATION

OF

DAVID L. COOPER

FOR

METHOD AND APPARATUS FOR FRACTAL COMPUTATION

[0001] This Application is a Continuation-In-Part of U.S. Patent Application Serial Number 09/240,052, filed January 29, 1999, which is a Continuation-In-Part of U.S. Patent Application Serial No. 08/713,470, filed September 13, 1996, now issued as U.S. Patent No. 6,009,418, which claims the benefit of U.S. Provisional Patent Application Serial No. 60/016,707 filed May 2, 1996. The entire disclosures of these applications, including references incorporated therein, are incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] The invention relates in general to the field of neural networks, and more particularly, to implicit general computation by means of digital applications or by means of dynamic systems with underlying fractal attractors.

Description of the Related Art

[0003] Neural networks of some description have been discussed in open literature since the 1960's. A few related patents (e.g., Dunn et al., U.S. Patent No. 3,275,986) date from that period. Despite much progress on the creation of distributed memory, robust function and pattern recognition, early work was largely abandoned after Marvin Minsky's famous attack on the Perceptron, which he showed to be incapable of the "exclusive or" (XOR) logical function. Interest in this type of computation device revived after the discovery that the XOR limitation could be overcome by adding additional layers to the Perceptron architecture, sometimes called "hidden" layers.

[0004] In the past 25 years, the U.S. has issued more than 4,000 patents for, or including components of, neural networks. The first of these (Cooper et al., U.S. Patent No. 3,950,733, and Cooper et al., U.S. Patent No. 4,044,243), a three layer design, claimed adaptive information

processing modules that mapped N input values to n output values, and thus involved neural networks performing general computation functions. However, since that time, only 11 percent of the U.S. patents related to neural networks concern general computation or generalized computational functions. More than 80 percent focus on specific applications employed on particular functions. Typically, these applications involve some kind of dynamic control (e.g., Grossberg et al., U.S. Patent No. 4,852,018), pattern recognition (e.g., Loris et al., U.S. Patent No. 4,876,731), or image or signal processing (e.g., Provence, U.S. Patent No. 4,885,757). The remaining patents concern components used in neural networks, either for general computation, or for specific applications (e.g., Kesselring, U.S. Patent No. 4,896,053).

[0005] Neural networks that perform either generalized or applied functions share a number of common traits, notably an architecture of computational nodes arrayed into one or more layers, connection weights, learning rules, and procedures to adjust parameters to expedite the learning process. More than 90 percent of these designs are supervised, meaning that the design requires a formal training phase where output values are "clamped" to a training set of data. Perceptrons provide an early example of such supervised networks and still provide useful applications (e.g., Adler, U.S. Patent No. 5,261,035). Perceptrons alone comprise more than 10 percent of the patented designs since 1976.

[0006] More than one third of all neural network design abstracts describe the use of backpropagation procedures. These provide an example of explicit computation in neural network functions, as backpropagation consists of employing the execution of an optimization algorithm to adjust the weights in a neural network architecture as the network is trained (e.g., Martin, U.S. Patent 5,440,651).

[0007] Backpropagation differs from feedback, which is simply the provision of inputs from other portions of an architecture, or from the environment. Backpropagation differs fundamentally from implicit computation, which occurs in networks that employ local rules to accomplish their tasks. Explicit computation in this sense describes the use of an external calculation to further the functioning of a network. The network in effect provides a framework in which the calculations, such as multi-variate Taylor expansions in the Cooper et al., designs above, are completed. Implicit computation does not need such external calculations. The contrast between implicit and explicit computation is quite similar to the distinction between simulation and emulation of dynamic systems discussed in David L. Cooper, *Linguistic Attractors,* Chapter 2 (1999): a simulation attempts to capture important aspects of dynamics, while an emulation attempts to match results without reference to internal dynamics (essentially a "black box" approach).

[0008] Two important classes of neural networks that normally rely on explicit calculation are the hidden Markov models and simulated annealing models. Hidden Markov models employ calculations based on a selected probability distribution to adjust the network as it trains. These distributions are stationary, that is, the probability of an event is the same at time t as at time $t + \Delta t$. For example, Brown et al., U.S. Patent No. 5,805,832, uses a hidden Markov step and a Poisson distribution for some applications. Abe, U.S. Patent 5,940,794, includes a hidden Markov step and mentions the gamma distribution in one embodiment (the gamma distribution corresponds to the distribution of waiting times for Poisson processes). Gong, U.S. Patent No. 6,151,573, uses a hidden Markov step with combinations of Gaussian (normal) distributions. Hidden Markov models account for more than 9 percent of the U.S. patents issued in the past quarter century.

[0009] Simulated annealing designs (e.g., Leuenberger, U.S. Patent No. 6,100,989)—at least another 6 percent of issued U.S. patents—are particularly suited to explicit calculation, as such designs incorporate a "temperature" parameter that adjusts the speed at which components change their weights. These are typically also associated with another probability distribution for which temperature is an important parameter: this is the Boltzmann distribution, which allows such designs to emulate thermodynamic systems. Implicit versions of simulated annealing are possible, for example, Alspector, U.S. Patent No. 4,874,963, implements the Boltzmann distribution with semi-conductor circuits, and uses a source of noise to adjust the "temperature" parameter.

[00010] Synthetic evolutionary designs comprise another 9 percent of issued U.S. patents. These (e.g. Parry et al., U.S. Patent No. 6,047,277) use a version of a "genetic algorithm" to produce random outputs, and then cull the outputs according to some metric. For example, Altshuler et al., U.S. Patent No. 5,729,794 uses such an algorithm to produce antenna designs, where computer estimates of antenna characteristics are weighed against a set of desired characteristics.

[00011] While neural network designs requiring explicit computation are very common, implicit designs, such as Alspector's cited above, are rare. Cooper, U.S. Patent No. 6,009,418, to which this application claims priority, is a clear example of this kind of design. It discloses an architecture that permits self-adjusting channels which already provides at least 26 percent improvement in digital simulations over other designs on deeply-nested dependency problems. It also incorporates learning rules based on non-stationary processes that permit non-digital implementations through dynamic systems characterized by such non-stationary processes, such

as systems described by Bose-Einstein statistics. Finally, it discloses a network design that can exploit the capability of fractal sets to encode and process information.

[00012] The present disclosure, in expanding on Cooper, 6,009,418, employs three key concepts that do not appear elsewhere in the prior art in the senses meant here: fractal sets, renormalization, and percolation. In the prior art, these terms are used in the following manner.

[00013] Except in Cooper, 6,009,418, "fractal" appears in three principal senses: as a method for data compression (e.g., Hoffberg et al., U.S. Patent No. 6,081,750), in the related sense in which it appears as an alternative method to re-construct a figure (e.g., Kostrzewski et al., U.S. Patent No. 6,167,155), and as a physical description, particularly as a texture (e.g., Nelson et al., U.S. Patent No. 6,052,485).

[00014] "Renormalization" occurs in the sense of a calculation step to bring values back into a specified range or re-scaling it (e.g., Gulati, U.S. Patent No. 6,142,681 and McCormack, U.S. Patent No. 5,265,192). In a minor exception, Barrett, U.S. Patent No. 5,602,964, notes that the disclosed process involving Liapunov exponents in that patent is "compatible" with renormalization group methods from statistical physics. Such methods are normally employed to derive gauge invariance in various systems.

[00015] "Percolation" occurs most often as a parameter a given design can compute as part of its output (e.g., Lewis et al., U.S. Patent No. 5,698,089). Bozich et al., U.S. Patent No. 5,367,612 uses "back percolation" in the sense of backpropagation. Colak, U.S. Patent No. 5,706,404, discloses a network design that uses inhomogeneities in a medium to transmit input signals as unchannelled waves. Colak comes closer to the sense employed in the present disclosure but stops short by using the percolation concept simply as a way to understand the process in that disclosure. The disclosure notes, for example, that there is no sharp cut-off in

current such as a real percolation model would predict. Klersy et al., U.S. Patent No. 5,536,947 describes a memory device that employs a material that changes back and forth between amorphous and crystalline states to store and retrieve information. They note that percolation takes place across the material in these switches. While memory is an important component to general computation, this disclosure does not take the next step and describe how such a process can be used to perform computations in general.

SUMMARY OF THE INVENTION

[00016] Accordingly, the present invention is directed to a method and apparatus for fractal computation that substantially obviates one or more of the problems due to limitations and disadvantages of the related art.

[00017] Additional features and advantages of the invention will be set forth in the description which follows, and in part will be apparent from the description, or may be learned by practice of the invention. The objectives and other advantages of the invention will be realized and attained by the structure particularly pointed out in the written description and claims hereof as well as the appended drawings.

[00018] To achieve these and other advantages and in accordance with the purpose of the present invention, as embodied and broadly described, in one aspect of the present invention there is provided a method for computation using a neural network architecture including the steps of using a plurality of layers, each layer including a plurality of computational nodes, an input processing layer, a central processing layer, and an output processing layer; using at least one feedforward channel for inputs, using full lateral and feedback connections within the processing layers, using an output channel for outputs, using re-entrant feedback from the output channel to at least one of the processing layers, using local update processes to update each of

the plurality of computational nodes, and using re-entrant feedback from the output channel to perform minimalization for general computation.

[00019] In another aspect of the present invention there is provided a apparatus for implicit computation including neural network architecture means having a plurality of layer means, each layer means including a plurality of adaptive computational node means, the plurality of layer means further including processing layer means including input processing layer means, central processing layer means, and output processing layer means, feedforward input channel means, full lateral and feedback connection means within the processing layer means, output channel means, re-entrant feedback means from the output channel means to the processing layer means, means for updating each of the plurality of adaptive computational node means using local update processes, and means for using re-entrant feedback from the output channel means to perform minimalization for general computation.

[00020] In another aspect of the present invention there is provided an apparatus for implicit computation including neural network architecture means including input means from an environment, and output means a plurality of locally connected computation node means for fractal percolation, wherein a minimalization step is used for computation.

[00021] It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory and are intended to provide further explanation of the invention as claimed.

BRIEF DESCRIPTION OF THE ATTACHED DRAWINGS

[00022] The foregoing and other objects, features, and advantages of the invention will be apparent from the following more particular description of preferred embodiments as illustrated in the accompanying drawings, in which reference characters refer to the same parts throughout

the various views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating principles of the invention.

[00023] In the drawings:

[00024] FIG. 1 illustrates a state space for a trajectory along a number line;

[00025] FIG. 2 illustrates a second and third steps in a construction of a top-down fractal percolation model (right) on a field of random inputs (left), with the probability of survival at each step set at 2/3;

[00026] FIG. 3 illustrates a remnant of the model in FIG. 2 after three steps;

[00027] FIG. 4 illustrates the remnant of a similar model after three steps when a probability of survival at each step is set at 0.85;

[00028] FIG. 5 illustrates a remnant of the set in FIG. 4 after two additional steps;

[00029] FIG. 6 illustrates a first three steps in a bottom-up process based on the same field of random inputs in FIG. 2, with the seed set based on a probability of selection of 0.15;

[00030] FIG. 7 illustrates a set in FIG. 6 after rule-driven extinction of elements not supported by the bottom-up percolation process;

[00031] FIG. 8 illustrates a result of an additional bottom-up step for a set in FIG. 7 after an additional extinction step;

[00032] FIG. 9 illustrates a result of feedback-driven rectification of the a in FIG. 8;

[00033] FIG. 10 illustrates first two steps in a second bottom-up cycle for remnants in FIG. 9;

[00034] FIG. 11 illustrates a result of a single extinction step and feedback-driven rectification of a set in FIG. 10;

[00035] FIG. 12 illustrates a first step in a third bottom-up cycle for remnants of sets from FIGs. 6-11;

[00036] FIG. 13 illustrates a diagrammatic layer view of the invention according to a preferred embodiment, including converging input channels, multiple processing layers, and diverging output channels for feedback or communication with other channels in the architecture;

[00037] FIG. 14 illustrates a schematic nodal view of elements within a self-organizing channel, including simultaneous output and feedback to the central processing portion of the channel;

[00038] FIG. 15a illustrates a schematic layer view of a 3-layer processing channel with feedforward output and feedback;

[00039] FIG. 15b illustrates a schematic layer view of a 3-layer processing channel with bi-directional output channels and feedback;

[00040] FIG. 16 illustrates a schematic layer view for an attention module;

[00041] FIG. 17 illustrates a schematic layer view for a long-term dynamic memory module;

[00042] FIG. 18 illustrates a schematic nodal view of a 3-layer processing channel with lateral connections to other channels and feedback from output channels;

[00043] FIG. 19 illustrates a textual or oral language parser with lateral processing connectivity to different channels;

[00044] FIG. 20 illustrates a basic file searcher with lateral connectivity to a single processing layer.

[00045] FIG. 21 illustrates an advanced file searcher with lateral connectivity to three central processing layers from selected parallel channels;

[00046] FIG. 22 illustrates an advanced file searcher capable of portraying propositional content of the files, with six central processing layers and selected lateral connectivity to parallel channels;

[00047] FIG. 23 illustrates a natural language translator with three central processing layers and selected lateral connectivity to parallel channels;

[00048] FIG. 24 illustrates a basic "smart" scanner with two central processing layers and selected lateral connectivity to parallel channels;

[00049] FIG. 25 illustrates an advanced "smart" scanner with six processing layers and selected lateral connectivity to parallel channels;

[00050] FIG. 26 illustrates a natural language dialect parser with four processing layers and selected lateral connectivity to parallel channels;

[00051] FIG. 27 illustrates an alternative channel structure for written language processing, with feedback from each channel to the first processing layer of parallel channels, and lateral connectivity among the second processing layers of the three channels;

[00052] FIG. 28 illustrates an alternative processing structure for written language processing similar to FIG. 27, which relies on self-organizing interlayer tessellation among the central processing layers; and

[00053] FIG. 30 illustrates an alternative channel structure for textual and oral natural language processing.

DETAILED DESCRIPTION OF THE PREFERRED EMDODIMENTS

[00054] Reference will now be made in detail to the preferred embodiments of the present invention, examples of which are illustrated in the accompanying drawings.

[00055] In a preferred embodiment, fractal computation is performed using a neural network architecture and learning rules derived from the study of dynamic systems, fractal percolation, and theory of general computation. Standardized components permit the storage, processing, and retrieval of information. The invention permits rapid, unsupervised processing of complex data sets, such as imagery, databases, textual files, or continuous human speech.

[00056] The standardized components of this invention, according to a preferred embodiment, include converging feedforward input channels, interlinked self-adjusting converging and diverging central processing channels, and output channels for processing execution instructions and for feedback. Learning rules are based on localized reinforcement of successes, as well as random or rule-driven inhibition. The learning rules incorporate features of non-stationary statistical processes such as Polya processes, Bose-Einstein statistical processes, phase transitions, or nearest-neighbor renormalization. These non-stationary processes provide the basis for computational non-linearities in digital applications, and for exploitation of new technologies based upon dynamic systems with underlying fractal attractors.

[00057] The term "fractal" was coined by Benoit Mandelbrot, who consciously avoided a fully rigorous definition of the term. For the purposes of this disclosure, and consistent with his discussion of the concept, a fractal is any set for which the Hausdorff dimension is greater than or equal to the topological dimension. Scaling or self-similar fractals form an important subset, which, when incorporated into iterated function systems, provide the basis for efficient data compression and decompression (see, for example, Barnsley et al., U.S. Patent Nos. 4,941,1931, 5,065,4471, and 5,347,600, and Barnsley, U.S. Patent No. 5,748,742). However, the feature

which makes iterated function systems ideal for data compression–high predictability–makes them unsuitable for computation, which requires encoding and manipulation of information. In fact, information which distinguishes one set from another is contained in a "residue" left over after iterated function systems have been employed to compress the data in the sets. For computation, this "residue" is often the most important component.

[00058] From the theory of Turing Machines, we know that computation can be broken into three essential components: composition, minimalization, and recursion. Composition associates with functions $f(n^m), g_1(r^n), g_2(r^n), g_3(r^n), \ldots, g_m(r^n)$ the function

[00059] $h(r^n) = f[g_1(r^n), g_2(r^n), \ldots, g_m(r^n)]$ where $(a_1, \ldots, a_n)$ is in the domain of each $g_i(r^n)$ $i = 1, 2, \ldots, m$, and $[g_1(a_1, \ldots, a_n), g_2(a_1, \ldots, a_n), \ldots, g_m(a_1, \ldots, a_n)]$ is in the domain of $f(n^m)$. The value of $h(r^n)$ at $(a_1, \ldots, a_n)$ is

[00060] $f[g_1(a_1, \ldots, a_n), g_2(a_1, \ldots, a_n), \ldots, g_m(a_1, \ldots, a_n)]$. Minimalization associates with each $f(y, r^n)$ the function $h(r^n)$ whose value for a given r^n is the least value of y, if it exists, for which $f(y, r^n) = 0$, and is undefined if it does not exist.

[00061] $h(r^n) = \min_y [f(y, r^n) = 0]$

[00062] Recursion is the repeated application of composition and minimalization.

[00063] The present invention permits dynamic computation on neural networks. Composition is assured by forward connectivity. Inputs are then automatically contained in any computation performed by any layer in the network. Recursion is assured by feedback. Minimalization is the non-trivial component in neural network computation. Most neural network designs rely on external computation to accomplish this step, particularly standard back-propagation designs, hidden Markov models, and so forth. Self-organizing designs, such as the Kohonen Quantum Learning Vectorization design (U.S. Patent No. 5,428,644), employ an

external computation to calculate the optimum value required by minimalization. Simulated annealing designs also rely on external computation, both of the local minimum or maximum, and for the artificial temperature parameter. The Semantic Attractor Architecture (U.S. Patent No. 6,009,418) accomplishes this step implicitly, relying on the characteristics of non-stationary dynamic processes. Of course, as is well known by those versed in the art, the Semantic Attractor Architecture can be simulated with digital representations of these processes as well.

[00064] Prior to assigning channel structures or unique characteristics at selected nodes, a neural network can be viewed as a lattice. Assume such a lattice of arbitrary area per layer and an arbitrary number of layers. Successful processes across the lattice can be viewed as a bond or site percolation process. For example, composition without recursion (feedforward) is equivalent to an oriented bond percolation setup across a network's layers. Bond percolation across lattices has a critical probability p_c such that a cluster is almost surely finite sized when $p < p_c$ and almost surely infinite and unique when $p > p_c$. For oriented percolation across lattices of dimension d,

[00065] $$p_c(d) \sim d^{-1}$$

[00066] With recursion, the dimension of the lattice is effectively infinite, unless truncated. Consider a trajectory from $-\infty$ to ∞ on a number line. The state space for this trajectory obviously requires 1 dimension, the number line itself. If the equivalent of a half-mirror is placed at the origin, however, where the trajectory continues with probability ½ or reverses with probability ½, the state space requires 2 dimensions to keep track of the potential location of the process on the number line. This corresponds to a single feedback link. To complete the feedback link as a recursive link, a second half-mirror is needed (assume for

simplicity that the half-mirror is transparent from the left). The second mirror requires a third dimension to portray leftward trajectories, illustrated at FIG. 1, but an infinite number of additive dimensions to portray oscillation between the half-mirrors, as well as the additional left and right trajectories. Thus, recursion, unless truncated by a time threshold, a limitation on iterations, or a decay process, leads to a lattice of infinite dimension. This effect carries forward as the line is replaced with planes (layers), and as recursion loops are incorporated into the network architecture.

[00067] Davis, The New Physics (1989), see pages 316-348, and particularly page 339, incorporated by reference herein, associates the time-dependent behavior of a self-organizing system with a macroscopic rate law and a deviation from a Poisson rate law. Prior to learning, a neural network represents a Poisson system with percolation probability $\gamma(\lambda)$ defined by $\gamma(\lambda) = \mathrm{P}^{\lambda}(|W| = \infty)$ where λ is the intensity of the Poisson process. There is a critical intensity λ_c such that

[00068] $$\gamma(\lambda) = \begin{cases} = 0 \text{ if } \lambda < \lambda_c \\ > 0 \text{ if } \lambda > \lambda_c \end{cases}$$

[00069] and the mean cluster size is E^{λ} (|W| is finite when $\lambda < \lambda_c$. In fact,

[00070] $$\zeta(\lambda) = \lim_{n \to \infty} \left\{ -\frac{1}{n} \log \mathrm{P}^{\lambda}(|W| = n) \right\}$$

[00071] exists and is strictly positive for $0 < \lambda < \lambda_c$. For $\gamma(\lambda) > 0$ there is almost surely a unique infinite cluster of overlapping spheres. The probability that any given vertex is open is given by

[00072] $$p_n(\lambda)=\mathrm{P}^{\lambda}(no\ \ point\ \ in\ \ ball\ \ \mathrm{B}_n(0)) = 1-\exp(-\lambda n^{-d})$$

[00073] where

[00074] $$B_n(x)=\prod_{i=1}^{d}\left[\, x_i-\frac{1}{2n}, x_i+\frac{1}{2n}\,\right) \ \ for\ x \in Z_n^d$$

[00075] Moreover, the critical probability

[00076] $$\lambda_c \geq -n^d \log\{1-p_c(L_n)\}$$

[00077] for lattice L with n nodes and d dimensions.

[00078] FIGs. 2 through 5 help to illustrate some of these points. FIG. 2 depicts two 100 x 100 pixel squares, with the square on the left consisting of a set of random inputs and the square on the right consisting of the remnant of those random numbers after three Poisson processes have been superimposed. The gray scale values in the left-hand square of random numbers go from white for probabilities between 0 and 0.25 in three additional steps of increasing gray values until the darkest gray shows probabilities between 0.75 and 1. To derive the right-hand figure from these values, three Poisson steps were superimposed, beginning with survival of probabilities greater than 1/3 on 25 x 25 pixel squares, then survival of probabilities greater than 1/3 on 5 x 5 pixels, and finally survival of probabilities greater than 1/3 on 3 x 3 pixel squares. The final two steps are depicted with black for the second step, and gray superimposed for the third. FIG. 3 shows the remnant surviving after the third step. These steps are typical for the construction of fractal sets in the mathematical literature, as in Falconer, Fractal Geometry (1990), see pages 231-235, and Meester and Roy, Continuum Percolation (1996), at pages 209-

216, both incorporated by reference herein. Clearly $p = 2/3$ is less than the critical probability for these processes.

[00079] FIG. 4 illustrates the remnant of a similar model after three steps when the probability of survival at each step is set at 0.85. FIG. 5 illustrates the remnant of the set in FIG. 4 after two additional steps, which are applied successively to 2 x 2 pixel squares, and finally to single pixels. In this case, percolation occurs from top to bottom, and nearly also occurs from left to right. This means that the critical probability for this set of inputs and these Poisson processes is close to 0.85.

[00080] Learning necessarily departs from the Poisson law, as the Poisson process only obtains to the point were the network's connection values are adjusted for success or failure. Derivation of the minimalization step is thus related to the macroscopic rate law required by Nicolis and Prigogine. (See Davis, The New Physics, pages 316-348.)

[00081] FIGs. 6 through 12 illustrate how implicit normalization can be used to accomplish this, and further demonstrate how construction of fractal sets on neural network lattices can accomplish computation.

[00082] Implicit normalization occurs when nodes behave in accordance with the behavior of the Γ nearest neighbors. In general, for a lattice with dimension d, N sites, and Γ nearest neighbors, the lattice is divided into blocks of length L_a, with a being the separation of the original lattice. Choose L so that $L_a << \xi, \xi$ being the correlation length of the lattice. The total number of nodes per block is L^d. The total number of blocks is NL^{-d}. The total on/off states in a Boolean block I is

[00083] $$S'_I = \sum_{i \in I} S_i$$

[00084] This can be expressed as a new variable

[00085] $S'_I = ZS_I \quad with \; S_i = \pm 1 \qquad Z = L^y$

[00086] Channels in a neural network architecture with standardized structures would be distinguished by their inputs and the intensity λ_c of the underlying Poisson process within the channel. When a solution develops in the channel, the path is unique. Updated connection or activation rates make the path more likely for identical inputs, up to $p = \lambda_c$.

[00087] For an example in two dimensions, FIG. 6 illustrates the first three steps in a bottom-up process based on the same field of random inputs in FIG. 2. The seed set is based on a probability of selection of 0.15, the converse of the top-down probability of 0.85 used for FIGs. 2 through 5. The steps used in FIG. 6 consist of completing any squares when two elements of the square are already occupied. The first step completes 2 x 2 squares, the second step completes 4 x 4 squares of the resultants of the first step, and the third step completes 6 x 6 squares using the new resultants.

[00088] For a further departure from the underlying random inputs and any Poisson process, FIG. 7 illustrates the set in FIG. 6 after the extinction of elements not supported by the bottom-up percolation process. Here, the elements were not supported whenever they were no greater than one pixel wide. This corresponds to local inhibition conditions often encountered in biological neural networks.

[00089] FIG. 8 illustrates the result of an additional bottom-up step for the set in FIG. 7 after an additional extinction step. Here the squares to be completed were 8 x 8 pixels, and extinction removed any elements less than two pixels wide.

[00090] FIG. 9 illustrates the result of feedback-driven rectification of the set in FIG. 8. By contrast to local inhibition, this corresponds to larger scale adjustments to network values, and thus requires feedback from later stages in processing. In FIG. 9, the adjustments transformed the underlying random values from x to $tanh(2x)$ for the percolating elements, retained the values for x for the larger "islands," reduced the values for x to $tanh(x)$ for the smaller "islands," and reduced the remaining values from x to $tanh(x/2)$. This simply introduces the non-linear adjustments to inter-nodal weights familiar to those versed in the art. In Boolean form, however, this step corresponds to the creation of an AND-NOT (NAND) gate, known to be sufficient for conventional computation when used in combination with other NAND gates.

[00091] Quantum computers can be built out of controlled-NOT (CNOT) gates. Percolation in a quantum context is an interesting area to explore, as many percolation studies involve atomic or molecular spin orientations, and percolation phenomena related to "Ising glasses." Spin orientations are quantum phenomena, and they can be manipulated in such simulations or experiments by use of radiation or the manipulation of the temperature. Leuenberger & Loss, Quantum Computing in Molecular Magnets, in Nature, vol. 410 (2001), pages 789-793, incorporated by a reference herein, disclose molecular magnets capable of dynamic random access memory, where the magnets are prepared and read out by multifrequency coherent magnetic radiation in the microwave and radio-frequency range. This article employs Grover's algorithm for quantum search of a database. As mentioned elsewhere in the present disclosure, memory is an important component of computation. The composition steps necessary for general computation in this case can be applied through manipulation of the signals that prepare network, much like the preparation in Leuenberger & Loss. Minimalization would occur at the read out step.

[00092] FIG. 10 illustrates the first two steps in a second bottom-up cycle for the remnants in FIG. 9. In this case, only two steps are required for percolation given the same underlying random inputs that began the process.

[00093] FIG. 11 illustrates the result of a single extinction step and feedback-driven rectification of the set in FIG. 10. Here, the underlying values were changed to *tanh(3x)* for the percolating cluster, *tanh(x/2)* for the smaller clusters, and *tanh(x/3)* for all remaining pixels. FIG. 12 illustrates the first step in a third bottom-up cycle for the remnants of the sets from FIG. 11. In the final figure, the network almost percolates immediately. Obviously, the cluster would percolate and no residues would be present for an extinction step after normalization with 2 x 2 squares.

[00094] Network dynamics depend on the creation of attractors within channels and the dynamic interaction of the attractors among channels. Attractors within channels arise by phase changes there. Conditions for their creation arise by re-entrant connections, which also increase the dimension of the network's state space. Dynamic interactions among channels occur by means of divergent and convergent connections, with bi-directional links also adding to the dimension of a network state space. Channels can evolve by means of self-organization within arbitrarily large layers, or by re-entrant lateral connections across standardized channels.

[00095] Most of these features are depicted in FIG. 13, which illustrates a diagrammatic layer view of the invention according to a preferred embodiment. It shows converging input channels, multiple bi-directional processing layers, and diverging output channels for feedback or communication with other channels in the architecture, as well as re-entrant connections to the processing layers to permit the creation of attractors. Divergent and convergent connections among channels are described below.

[00096] The layers in FIG. 13 consist of processing nodes. In subsequent diagrams, both the layer view here or a nodal view will appear, as appropriate. Unless the diagram ascribes a particular function to a layer or node, or unless it specifies the number of elements, the diagram is meant to be suggestive. That is, an architecture, if required, could have more or fewer elements of a given type.

[00097] FIG. 14 illustrates a schematic nodal view of the elements within a self-organizing channel, including simultaneous output and re-entrant feedback to the central processing portion of the channel. Input processing consists of convergent, feedforward connections to the processing portion of the architecture. Outputs go as necessary to other channels or memory storage, and may also be used as the inputs for programming a second neural network.

[00098] FIG. 15 shows two potential variants for standardized processing channels. FIG. 15a illustrates a schematic layer view of a 3-layer processing channel with feedforward output and feedback. Input and output layers are depicted in gray, and may be implemented by more than the one layer shown. This diagram shows full feedback connectivity among the processing layers, and re-entrant feedback to the processing layers from the output layer.

[00099] FIG. 15b illustrates a schematic layer view of a 3-layer processing channel with bi-directional output channels and feedback. This kind of standardized component would accommodate related outputs that do not interact directly. For example, in language processing, to distinguish between the senses of the word "bat," which can be an animate noun, an inanimate noun or a verb, would not lend itself to a single look-up step. This is particularly the case when the verb can further be subdivided into actions which may or may not require wooden or metallic tubular objects to execute them, and the animate noun can be subdivided into a wide variety of

species. Moreover, the word can appear as an adjective in such phrases as "bat cave." In such cases the final processing layer acts as a hidden intermediary between the output channels. The representations of the word to not interact directly. FIG. 15b also depicts re-entrant feedback to the processing layers and full feedback connectivity within the processing component of the channel.

[000100] FIG. 16 illustrates a schematic layer view for an attention module. Such a module would create bi-directional connections to processing elements of the channels in a network architecture, and would permit the top-down processing capable of amplifying the rectification of patterns shown in FIGs. 9 and 11.

[000101] Neural network architecture can employ standard digital memory as input. Dynamic memory that evokes previous cross-channel states in the architecture is also possible. FIG. 17 illustrates a schematic layer view for a long-term dynamic memory module with bi-directional connectivity to an attention module and any processing channels in a neural network architecture. The storage layers are principally feed-forward to encourage a sparse coding action that is relatively stable over time.

[000102] FIG. 18 provides an alternate schematic nodal view of a 3-layer processing channel with lateral connections to other channels and feedback from output channels. This describes input to each layer from parallel channels which may be desirable. Re-entrant feedback is also depicted. The "reset" feature is meant to depict the option of input from an attention module. Full feedback connectivity within the central processing layers is also shown.

[000103] The obvious parameter to provide a minimalization value for random percolation within the channels of a neural network architecture is the critical probability, or the critical intensity when the channel begins with a Poisson process. Above this value, the channel

percolates, below it, it does not almost surely. Accordingly, a neural network architecture can be optimized for learning tasks by adjusting the dimensionality of the network and the connectivity weights so that it begins learning below the critical parameter. For rapid percolation along a channel, the optimal initial values would be below the critical parameter, but where the distance to that parameter is not large. Conversely, for interactions among the elements in a processing layer, where percolation would presumably not be optimal, connection weights should be well below the critical parameter, thus ensuring sparse coding within the layer. The role of lateral inhibition would be particularly important in this regard, as it would also help to confine percolation along the channel and prevent percolation across it.

[000104] The critical intensity or critical probability of a network architecture depends on its geometry, the connectivity of its elements between layers and across layers, and the time a given input affects the network. As noted before, re-entrant and feedback connections increase the dimensionality of the network in state space. Inhibition and decay of input signals prevent the dimensionality from increasing to infinity. Various forms for a decay function were discussed in Cooper, U.S. Patent No. 6,009,418 in connection with learning rules derived from a Polya process, which is a non-stationary random process that can serve as an alternative to nearest-neighbor normalization discussed above. For example, to model "forgetting," which avoids saturation of the synaptic weights, the Polya rules can be modified by introducing a decay factor. Since the decay factor is not related to activity, but simply to time, any node that is inactive will eventually approach a connectivity of zero, for any non-zero decay rate. For connection weight w_{ji} from node i to node j, initial weight $\left|\frac{r}{s}\right| \leq 1$, number of excitations α and number of inhibitions β, decay at a node can then be modeled by:

[000105] $$w_{jt}(t)=\frac{r+a-\beta}{s+\eta+\delta(t)}$$

[000106] for $\alpha+\beta=\eta$ *all positive;* $\delta(t)\geq 0$ *monotone increasing*

[000107] Other decay rates are possible. For example, the common form

[000108] $$w_{jt}(t)=\frac{r+\alpha-\beta}{s+\eta}-\delta(t),\ \delta(t)\geq 0$$
$$e.g.\ \delta(t)=e^{-kt}\left(\frac{r+\alpha-\beta}{s+\eta}\right)$$

[000109] A number of methods are available to estimate the critical probability or critical intensity for a network. Consider continuum percolation on a unit hypercube with dimension *d*. That is, the number of elements in a layer and the number of layers is arbitrarily large, and the network has sufficient re-entrant connections to provide a state space of dimension *d*. FIGs. 2-12 represent such a process for two dimensions assuming the individual pixels are reduced to zero area. For a Poisson process (X_k, ρ_k, λ_k), where X_k characterizes the process, ρ_k is the diameter, and λ_k is the intensity at step k, let $\rho_k=a\rho$ with $a=m^{1-k}$, $\lambda_k=b\lambda$ with $b=m^{-d(1-k)}=m^{d(k-1)}$ for side m^{-1} and step k. Let

[000110] $$V_\infty=\bigcap_{k-1}^{\infty}V_k$$

[000111] be the vacant sections of the unit hypercube. *P(x = 0)* is independent at each step with

[000112] $$P(0)=\exp\left(-\lambda/m^d\right)$$

[000113] $V_\infty=\emptyset$ almost surely when $m^d\exp\left(-\lambda/m^d\right)<1$ so

$$\lambda>m^d\log\left(m^d\right)=dm^d\log m$$

[000114] $$\dim V_\infty = \log\left[m^d \exp\left(-\lambda / m^d\right)\right] \log m = d - \frac{\lambda}{m^d \log m}$$

[000115] Conversely,

[000116] $$P(x \geq 0) \geq \exp\left(-m^{d(3-1)} \lambda \left[1 - 1/m^d\right]\right) \exp\left(-m^{d(4-1)} \lambda \left[1 - 1/m^d\right]\right)$$
$$\geq \exp\left(-\lambda m^d \left[m^{2d} - 1\right]\right)$$

[000117] for each corner.

[000118] $V\infty \neq \varnothing$ almost surely when $m^d \exp\left(-\lambda m^d \left[m^{2d} - 1\right]\right) > 1$ so

[000119] $$\lambda < \log\left(m^d\right) / m^d \left(m^{2d} - 1\right)$$

[000120] and

[000121] $$\dim V_\infty = d - \frac{\lambda m^d}{\log m}\left(m^{2d} - 1\right)$$

[000122] Let $\theta_f(\lambda)$ be the probability that $V_\infty = \cap[0, 1]^d$ contains a connected component which intersects the opposite side of $[0, 1]^d$. Define λ_f as

[000123] $$\lambda_f = \inf\{\lambda : \theta_f(\lambda) = 0\}$$

[000124] Then $\lambda_f > 0$ and $\lim_{n\to\infty} \lambda_f(n) = \lambda_c^*(1) = \lambda_c(1)$ so the critical parameter for percolation is the same as the critical fractal parameter, and marks the threshold where percolation occurs almost surely.

[000125] Similar results obtain for generalized Cantor sets applied to fractal percolation. A statistically self-similar random Cantor set is constructed as follows. $F = \bigcap_{k=1}^{\infty} E_k$ for $[0,1] = E_0 \supset E_1 \supset \ldots$ With E_k a union of 2^k disjoint basic intervals. For left/right intervals I_L / I_R of E_{k+1}, $|I_L|/|I|$ and $|I_R|/|I|$ have the same probability distribution. This random Cantor set has $\dim_H F = s$ such that $E\left(C_1^S + C_2^S \ldots\right) = 1$ for constants $C_{i1,\ldots,\,ik} = |I_{i1},\ldots,I_{ik}|/|I_{i1},\ldots,I_{ik-1}|$ for $a \le C_{i1,\ldots,ik} \le b$ and for all $i_1,\ldots,i_k$ and both L and R.

[000126] The probability *q*, that *F* is empty, is the smaller non-negative root of the polynomial

[000127] $$f(t) \equiv \sum_{j=0}^{m} \mathrm{P}(N = j) t^j = t$$

[000128] With probability *1 - q* the set *F* has Hausdorff and box dimension given by the solution of $E\left(\sum_{j=0}^{n} C_i^S\right) = 1$. For a middle *1/m* construction, and *N* the random number of hypercubes in $\Re^d$ of E,

[000129] $$\mathrm{P}(N = j) = \binom{m^d}{j} p^j (1-p)^{m^d - j} t^j = (p\,t + 1 - p)^{m^d} = q$$

[000130] and

[000131] $$E\left(\sum_{j=0}^{m^d} C_1^S\right) = 1 \rightarrow E\left(\sum_{j=0}^{m^d} m^{-S}\right) = m^{-S} E(N) = m^{-S}\left(m^d\right)\mathrm{P}$$

[000132] so

[000133]
$$\dim_H F_\rho = \frac{\log m^d \mathrm{P}}{\log m} = d\left(\frac{\log \mathrm{P}}{\log m} + 1\right)$$

[000134] When p is determined by a Poisson process with number of trials n and intensity λ

[000135]
$$\dim_H F_{\mathrm{P}} = d\left(\frac{\log n - \log \lambda}{\log m} + 1\right)$$

[000136] It is also possible to estimate the critical parameters empirically. For example, Okabe et al., Spatial Tessellations, (2000), at pages 361-362, incorporated by a reference herein, produced the following threshold probabilities for bond and site percolation models on Poisson Voronoi diagrams and Poisson Delaunay tessellations in $\Re^m$:

		Site	Bond
m = 2	Delaunay	0.5	2 sin (π/18)
	Voronoi	0.5	0.332
m = 3	Voronoi	0.1453	0.0822

[000137] Given a set of distinct, isolated points in a continuous space, Voronoi diagrams associate the nearest regions of the space with each point. Delaunay tessellations can be constructed from Voronoi diagrams in m-dimensional space by joining points whose regions share an (m - 1)-dimensional face. These are important tools in analyzing a self-organizing neural network architecture. To see this, assume two layers in an architecture are connected by overlapping projections (tessellations), so that a node on layer i can influence the projections of one or more of its neighbors on layer j. As the network evolves, these projections will be strengthened or weakened, depending on the interaction of the network with its environment.

Eventually, competition among the nodes on layer *i* will exclude the influence from neighboring nodes on some portion of layer *j* for each node, thereby creating regions that can be captured in Voronoi diagrams or the associated Delaunay tessellation. This is an example of implicit calculation through the interaction of network elements, and how channels can organize themselves as a network evolves.

[000138] Thus, it is possible to harness locally interacting nodes in a neural network so that their inherent characteristics permit percolation across the network. Inter-nodal interactions provide the means to evolve self-organizing channels, as well as the feedforward properties necessary for composition. Re-entrant and feedback connections provide the necessary recursive properties for computation. Finally, critical percolation parameters provide the necessary factors for implicit minimalization within the network. When further interaction with the environment rectifies these factors further, percolation permits even more rapid learning and adaptation. Thus, all three conditions are present for implicit general computation.

[000139] These elements are present when the components of the network provide the properties for fractal percolation, regardless of their physical composition. Thus, digital implementation on semiconductor chips is not required. For example, as discussed in Cooper, U.S. Patent No. 6,009,418, the Polya rules mentioned above already accommodate implementation by means which demonstrate Bose-Einstein statistics. Lasers fall into this category. This discussion goes further to describe how any dynamical system with non-stationary behavior can be employed. This is particularly true for systems where such non-stationary behavior amounts to a thermodynamic phase change. Consequently, this invention can be implemented in quantum embodiments when channels can undergo Bose-Einstein condensation, for example.

[000140] Computation theory teaches that the converse case is also possible. The output of any computation system can be duplicated by Turing machines provided with the necessary "prefix" that describes that system. In this sense, the present disclosure provides a framework for digital emulation of non-stationary elements linked together in a neural network. Such emulations can take advantage of known dynamical characteristics to enhance computation performance, which has already been accomplished in trials of Cooper, U.S. Patent No. 6,009,418, which provided 26 percent improvement in deep-nested tree problems over hybrid backpropagation designs that had been optimized for particular problem sets.

[000141] This disclosure also offers a useful feature in allowing the development of standard components that can be linked together to solve particular computational problems. Such linkages can be virtual, using digital means on existing computers, or physical, using potentially much faster arrays of dynamic elements. The discussion of critical parameters above revealed some of the principles that would make this possible: modification of the dimensionality of the network by varying re-entrant and feedback connections, use of inhibition and decay to input signals, and adjustment of probabilities so that elements in a channel are either close to or far from the critical thresholds, as required. Moreover, the "leaky" processing described in Cooper, U.S. Patent No. 6,009,418 and the overlapping tessellations mentioned in the discussion of Voronoi and Delaunay portrayals of *m*-dimensional spaces provide the necessary mechanism for standard-size channels to alloy larger layer arrangements when necessary: outputs to neighboring standardized channels spillover competition in neighboring channels, and thus the adaptive capability for a channel attractor to extend its influence over nodes not originally part of the standard channel structure.

[000142] These ideas permit a broad number of applications, a fraction of which are outlined below.

[000143] FIG. 19 illustrates a textual or oral language parser with lateral processing connectivity to different channels. The parser receives oral or written input separated into words. Oral inputs received as features or formants would require the more complex architecture at FIG. 25. The words proceed to the next layer, where the network accesses a glossary or dictionary, which labels the words by type. Lateral feedback at this level would suppress impossible combinations. At the next layer, the network matches each adjacent pair of words to possible templates, such as *Determiner + Noun*. Feedback to the previous layer would suppress further impossible combinations, and reinforce matches. Lateral feedback would also do this. At the third processing layer, the network would access clausal templates against the remaining pairs of words, again suppressing impossibilities and reinforcing matches. This layer would be capable of checking for word-order clues. Feedback from this layer to the first processing layer can check for verb agreement and other such characteristics, thereby reducing the number of remaining options, and reinforcing the rest. At the fourth processing layer, the network accesses sentence templates, which allows it to check for textual relations to other sentences, and to correlate portions of compound, complex, conditional, and other sentence types. Finally, the network resets itself for the next sentence and produces propositional outputs and any desired statistical data. At this level, the parser would be capable of reducing the input text to its logical propositional components (e.g., 'if a then b,' 'either a or b,' 'both a and b,' 'a but not b,' etc.), as well as to basic syntactic components *V(X, Y, Z) + A*, where 'V' is the verbal component of each clause, 'X' the focus, 'Y' the target (if any), 'Z' the reference (if any), and 'A' is the set of all

adjuncts appended to each clause. These concepts and notations are discussed at greater length in Cooper, *Linguistic Attractors,* Chapter 5 (1999) incorporated by reference herein.

[000144] The file searcher designs in FIGs. 20-22 are related to the parser in two ways. First, they share the architectural features mentioned above, with more and more layers added as more and more advanced features are added. Second, they rely on parser outputs as templates for the search. In FIG. 20, the basic search design relies on X- or V- template inputs from the parser. The X- template would produce a noun index related to the focus (typically the subject) noun phrases in a basic text, thereby creating relatively sophisticated search parameters simply by parsing a base document of interest. Similarly, the V-template would employ the verbs as the basis for search. The more advanced search architecture at FIG. 21 allows more processing of the searched files before employing the parser inputs. The searcher would then match X- or V-templates from the base document against comparable arrays in the searched files, thereby creating an index of matching clausal parameters, rather than simple nouns or verbs. The most sophisticated of these searchers is at FIG. 22, which would match entire propositional outputs from the base document and all searched files. See Cooper, Lingustic Attractors (1999) for further discussion of the basic concepts.

[000145] FIG. 23 applies this approach to language translation. It takes the propositional output from a parsed document (e.g., from FIG. 19) or a parsed oral expression (e.g., from FIG. 25) in one language. It then looks up the words for matches in a second language, adjusts the alternatives against clausal templates of the second language, and finally against full sentence templates. The output would be an expression in the second language with the same propositional content as the input expression.

[000146] FIG. 24 represents a "smart" scanner, which takes inputs from a scanning device, matches the scanned characters against stored font templates, and adjusts the results based on a subsequent retrieval of dictionary information. As with the searcher design, smart scanners can employ any of the lookup/template levels from the parser architecture to adjust the final estimate of what has been scanned.

[000147] The most sophisticated scanner is represented at FIG. 25, which brings sentence templates to bear before producing an output. This architecture would also work for oral input, with phoneme data rather than character data.

[000148] FIG. 26 shows a design to help calibrate verbal signals. It takes oral inputs as processed into features or formants, then adjusts those estimates for the phonemic environment, for the estimated age and gender of the speaker, and then for regional (dialect) effects. The final output is in standard phonemes, which can be employed by the advanced parser at FIG. 25.

[000149] As the foregoing discussion implies, these channel structures can be combined. FIG. 27 illustrates an alternative channel structure to FIGs. 19-22 for written language processing that employs both standardized channel components from FIG. 15. The channel designs in FIGs. 25 and 26 could be employed equally well. FIG. 27 also depicts feedback from each channel to the first processing layer of parallel channels, and lateral connectivity among the second processing layers of the three channels. Consequently, this figure also shows how some of the word and sentence templates in previous figures could be developed in a single network. The roots/words layer screens for word templates, similar to a dictionary lookup. This directly feeds a semantics layer, but indirectly influences channels for syntax and phonemics. Similarly, the layer for screening sentences feeds the layer for syntactic templates ("Syntax/Gestalt"), and the layer for orthographic rules feeds the layer for phonetic representations. Unlabeled layers

provide additional processing, normally associated with "hidden" layers in the art. The bi-directional outputs to the role and filler layers provide the processing for ambiguous elements such as the word "bat" described earlier in the disclosure.

[000150] FIG. 28 illustrates an alternative processing structure for written language processing similar to FIG. 27, which relies on self-organizing interlayer tessellation among the central processing layers. This embodiment would permit freer self-organization than the design in FIG. 27, but at the expense of longer learning times.

[000151] FIG. 30 illustrates an alternative channel structure for oral or textual natural language processing, with an attention module and a dynamic long-term memory module. Bold lines are bi-directional. The dashed lines are feedforward. The attention module is connected to the bottom two processing layers of the connected channels. Long-term memory is linked to a single processing layer, normally the first or second. As with FIG. 27, feedback for the remaining modules is through the first processing layer in the connected parallel channels; lateral connectivity is provided among the second processing layers. Connections to the dynamic long-term memory module are through the first layer.

[000152] FIGs. 18, 19, 21-23, 25, and 26 illustrate alternatives where more than two layers of parallel channels possess lateral connections.

[000153] While the invention has been described in detail and with reference to specific embodiments thereof, it will be apparent to those skilled in the art that various changes and modifications can be made therein without departing from the spirit and scope thereof. Thus, it is intended that the present invention cover the modifications and variations of this invention provided they come within the scope of the appended claims and their equivalents.